Name: Shahriar Ahmed

ID: 20101588

Section: 08

Lab Assignment: 06

#####Task-1:

inputTxt = open("input.txt", "r")

outputTxt = open("output.txt", "w")

file = inputTxt.readlines()

noProcesses = file.pop(0)

noResources = file.pop(0)

maxMatrix = []

allocationMatrix = []

a = ''

b = file.pop()

a = a + b

c = []

for i in range(0, len(a)):

c.append(a[i])

resource = []

for i in range(0, len(c), 2):

resource.append(int(c[i]))

A1, B1, C1, D1 = file.pop(0), file.pop(0), file.pop(0), file.pop(0)

A2, B2, C2, D2 = file.pop(0), file.pop(0), file.pop(0), file.pop(0)

a1, a2, b1, b2, c1, c2, d1, d2 = [], [], [], [], [], [], [], []

for i in range(0, len(A1)):

a1.append(A1[i])

for i in range(0, len(B1)):

b1.append(B1[i])

for i in range(0, len(C1)):

c1.append(C1[i])

for i in range(0, len(D1)):

d1.append(D1[i])

for i in range(0, len(A2)):

a2.append(A2[i])

for i in range(0, len(B2)):

b2.append(B2[i])

for i in range(0, len(C2)):

c2.append(C2[i])

for i in range(0, len(D2)):

d2.append(D2[i])

d3, d4, d5, d6, d7, d8, d9, d10 = [], [], [], [], [], [], [], []

for i in range(0, len(a1), 2):

d3.append(int(a1[i]))

for i in range(0, len(b1), 2):

d4.append(int(b1[i]))

for i in range(0, len(c1), 2):

d5.append(int(c1[i]))

for i in range(0, len(d1), 2):

d6.append(int(d1[i]))

for i in range(0, len(a2), 2):

d7.append(int(a2[i]))

for i in range(0, len(b2), 2):

d8.append(int(b2[i]))

for i in range(0, len(c2), 2):

d9.append(int(c2[i]))

for i in range(0, len(d2), 2):

d10.append(int(d2[i]))

maxMatrix.append(d3)

maxMatrix.append(d4)

maxMatrix.append(d5)

maxMatrix.append(d6)

allocationMatrix.append(d7)

allocationMatrix.append(d8)

allocationMatrix.append(d9)

allocationMatrix.append(d10)

class Bankers:

def \_\_init\_\_(self):

self.needMatrix = []

self.changeResource = []

self.safeSequence = []

def needMatrixUp(self, maxMatrix, allocationMatrix):

rows = 4

cols = 5

for i in range(0, rows):

for j in range(0, cols):

self.needMatrix.append(maxMatrix[i][j] - allocationMatrix[i][j])

return self.needMatrix

def safeSequenceUp(self, newNeed, available, allocationMatrix, noProcesses, noResources):

running = [True] \* noProcesses

count = noProcesses

while count != 0:

safe = False

for i in range(noProcesses):

if running[i]:

executing = True

for j in range(noResources):

if newNeed[i][j] - allocationMatrix[i][j] > available[j]:

executing = False

break

if executing:

outputTxt.write(str("Executing Process :\n"))

if int(i+1) == 1:

outputTxt.write(str("A"))

elif int(i+1) == 2:

outputTxt.write(str("B"))

elif int(i+1) == 3:

outputTxt.write(str("C"))

elif int(i+1) == 4:

outputTxt.write(str("D"))

running[i] = False

count -= 1

safe = True

for j in range(noResources):

available[j] += allocationMatrix[i][j]

break

if not safe:

outputTxt.write(str("\nthe processes are in an unsafe state."))

break

outputTxt.write(str("\nThe process is in a safe state & available resources : "))

for j in range(len(available)):

outputTxt.write(str(available[j]))

outputTxt.write(str("\n"))

outputTxt.write(str("\n"))

a = Bankers()

updated = a.needMatrixUp(maxMatrix, allocationMatrix)

newNeed = []

x1, x2, x3, x4 = [], [], [], []

for i in range(0,len(updated)-15):

x1.append(updated[i])

for i in range(5,len(updated)-10):

x2.append(updated[i])

for i in range(10,len(updated)-5):

x3.append(updated[i])

for i in range(15,len(updated)):

x4.append(updated[i])

newNeed.append(x1)

newNeed.append(x2)

newNeed.append(x3)

newNeed.append(x4)

available = resource

outputTxt.write(str("Need Matrix :\n"))

for i in range(len(newNeed)):

for j in range(5):

outputTxt.write(str(newNeed[i][j]))

outputTxt.write(str("\n"))

outputTxt.write(str("\n"))

outputTxt.write(str("Safe Sequence is & Change in available resource matrix :\n"))

outputTxt.write(str("\n"))

a.safeSequenceUp(newNeed, available, allocationMatrix, 4, 5)